Refine Search

Search Results -

Terms	Documents
709/227	4507

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Tuesday, November 30, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L20	709/227	4507	L20
L19	709/224	6364	L19
L18	L17 and (mobile near computer or satellite near dish)	35	L18
L17	L16 and (secondary or second) near stor$	569	L17
L16	L15 and (primary or first or main) near stor$	1315	L16
L15	L14 and (data with base or database)	11894	L15
L14	L13 and client	17291	L14
L13	L12 and (www or network or internet)	30677	L13
L12	709.clas.	32491	L12
L11	L8 and unused near (partition or space or section)	70	L11
L10	L8 and (partition or space or section)	4830	L10
L9	L8 and (partition or space)	3876	L9
L8	L7 and (data near record or data near file)	6747	L8
L7	L6 and (data with base or database) near manag$	24760	L7

L6	(garbage near 3 manag$ or trash near manag$)	592124	L6
L5	garbage near 3 manag$	592124	L5
L4	707.clas.	23648	L4
L3	707/204	2033	L3
L2	707/205	1798	L2
L1	707/206	1036	L1

END OF SEARCH HISTORY

L11: Entry 55 of 70 File: USPT Nov 23, 1999

US-PAT-NO: 5991776
DOCUMENT-IDENTIFIER: US 5991776 A

TITLE: Database system with improved methods for storing free-form data objects of data records

DATE-ISSUED: November 23, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bennett; John Grant	San Mateo	CA		
Shaughnessy; Steven T.	Scotts Valley	CA		
Brumme; Christopher Wellington	Boulder Creek	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Inprise Corporation	Scotts Valley	CA			02

APPL-NO: 08/ 667575 [PALM]
DATE FILED: June 21, 1996

PARENT-CASE:
This is a divisional patent application of Ser. No. 08/109,033 filed Aug. 18, 1993 now U.S. Pat. No. 5,561,793, which is itself a continuation-in-part application of application Ser. No. 07/933,480, filed Aug. 20, 1992, now U.S. Pat. No. 5,555,388 the disclosure of which is incorporated herein by reference.

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/205; 707/100
US-CL-CURRENT: 707/205; 707/100

FIELD-OF-SEARCH: 395/600, 707/200, 707/205, 707/206, 707/1, 707/100

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4429372	January 1984	Berry et al.	707/508
☐	4716404	December 1987	Tabata et al.	340/723

☐	4748678	May 1988	Takeda et al.	382/56
☐	4893232	January 1990	Shimaoka et al.	364/200
☐	4912640	March 1990	Tsugei	364/400
☐	5063501	November 1991	Jordan, Jr.	395/725
☐	5109336	April 1992	Guenther et al.	395/497.02
☐	5109508	April 1992	Mitsumori et al.	395/600
☐	5159678	October 1992	Wengelski et al.	395/480
☐	5214779	May 1993	Barker et al.	395/200.66
☐	5218539	June 1993	Elphick et al.	707/531
☐	5239466	August 1993	Morgan et al.	395/148
☐	5269019	December 1993	Peterson et al.	707/205
☐	5341466	August 1994	Perlin et al.	395/139
☐	5404435	April 1995	Rosenbaum	395/777
☐	5481645	January 1996	Bertino et al.	395/2.79
☐	5490260	February 1996	Miller et al.	395/427
☐	5546557	August 1996	Allen et al.	395/438
☐	5557794	September 1996	Matsunaga et al.	395/600
☐	5615367	March 1997	Bennett et al.	395/613

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
2077949	March 1990	JP	

OTHER PUBLICATIONS

dBase IV for Developers, Programming with dBase IV, Ashton-Tate Corporation, 1988, 1990, pp. 3-9 to 3-12.
Language Reference, Appendix E: Structure of a Database (.dbf) File, Ashton-Tate Corporation, 1988, 1990, pp. E-1 to E-4.
ReFlex User's Guide, Chapter 1: Creating and Modifying a Database, Borland International, Inc., 1984, 1989, pp. 7-36.
Townsend, C., Mastering dBase IV Programming, Chapter 20: Using Memo Fields, Sybex, Inc., 1989, pp. 331-342.
Lock Management Architecture, IBM Technical Disclosure Bulletin (1989) 31:125-128.
Conditional Locking of Nonroot Index Pages, IBM Technical Disclosure Bulletin (1989) 32:57-58.
Processor for Distributed Cross System Locks, IBM Technical Disclosure Bulletin (1978) 20:4760-4762.

ART-UNIT: 277

PRIMARY-EXAMINER: Von Buhr; Maria N.

ATTY-AGENT-FIRM: Smart; John A.

ABSTRACT:

A system of the present invention includes a relational database management system (RDBMS). Methods are described for maintaining integrity between "design documents," which may be creating under different operating systems, and one or more information tables of the system. The system provides each field of a table with a unique ID ("field ID") for tracking the field regardless of restructuring changes which may be made to the table by various clients. Corresponding field IDs are stored with the fields of design documents, thereby permitting the system to maintain a link between a design document and its table. Upon a restructure of a table, the dependent design documents may be appropriately updated by their respective clients. Methods are also described for improved storage of free-form or "memo" data. In a preferred embodiment, memo data are stored in a separate file comprised of variable-length storage blocks. Methods are described for allocating storage space in the blocks and sub-allocating storage space within a block. For increased efficiency, the system maintains a sorted "free list" of free storage blocks.

34 Claims, 33 Drawing figures

Previous Doc Next Doc Go to Doc#

L11: Entry 55 of 70 File: USPT Nov 23, 1999

DOCUMENT-IDENTIFIER: US 5991776 A
TITLE: Database system with improved methods for storing free-form data objects of data records

Abstract Text (1):
A system of the present invention includes a relational database management system (RDBMS). Methods are described for maintaining integrity between "design documents," which may be creating under different operating systems, and one or more information tables of the system. The system provides each field of a table with a unique ID ("field ID") for tracking the field regardless of restructuring changes which may be made to the table by various clients. Corresponding field IDs are stored with the fields of design documents, thereby permitting the system to maintain a link between a design document and its table. Upon a restructure of a table, the dependent design documents may be appropriately updated by their respective clients. Methods are also described for improved storage of free-form or "memo" data. In a preferred embodiment, memo data are stored in a separate file comprised of variable-length storage blocks. Methods are described for allocating storage space in the blocks and sub-allocating storage space within a block. For increased efficiency, the system maintains a sorted "free list" of free storage blocks.

Brief Summary Text (6):
The present invention relates generally to information processing environments and, more particularly, to storing, retrieving, and presenting information in a data processing system, such as a Database Management System (DBMS).

Brief Summary Text (7):
Computers are a powerful tool for the acquisition and processing of information. Computerized databases can be regarded as a kind of electronic filing cabinet or repository for collecting computerized data files; they are particularly adept at processing vast amounts of information quickly. As such, these systems serve to maintain information in database files or tables and make that information available on demand. Of these systems, ones which are of particular interest to the present invention are Relational Database Management Systems (RDBMSs).

Brief Summary Text (12):
The first of these, structure, is how data should be presented to users. A database management system is defined as "relational" when it is able to support a relational view of data. This means that data which a user can access and the operators which the user can use to operate upon that data are themselves relational. Data are organized as relations in a mathematical sense, with operators existing to accept relations as input and produce relations as output. Relations are perhaps best interpreted by users as tables, composed of rows (tuples) and columns (attributes).

Brief Summary Text (16):
The general construction and operation of a database management system is known in the art. See e.g., Date, C., An Introduction to Database Systems, Volumes I and II, Addison Wesley, 1990; the disclosures of which are hereby incorporated by reference.

Brief Summary Text (17):
Today, relational systems are everywhere--commonly seen operating in corporate, government, academic settings, and other shared environments. With the movement of data processing chores from mainframe computers to networked desktop computers, a particular problem has arisen however. Often a company's data will be maintained in information tables on one system but viewed in forms and reports of other systems. For instance, a company may maintain sales data on a file server operating under Novell NetWare on the one hand, with individual users viewing that information in various forms and reports at client workstations operating under disparate operating systems (e.g., MS/PC-DOS Windows, Macintosh, and the like) on the other hand. As a result, discrepancies between the information tables and their clients may occur. If one client modifies the structure of a table, for instance, the forms and reports of other clients which are dependent on that table may be rendered inconsistent (with the table) or even invalid.

Brief Summary Text (19):
Prior art approaches to storing this free-form or "memo" data have included so-called memo files employing fixed-length storage blocks. In dBASE III.RTM., for instance, a table of database records would store memo information in an accompanying memo file comprising 512-byte storage blocks. The approach is very wasteful: a record having only 40 bytes of memo information would require as much storage space as one having 500 bytes. Moreover, such conventional systems include no free-space management which would allow reclamation of storage space which has been freed (e.g., after its corresponding database table record has been deleted).

Brief Summary Text (21):
A system of the present invention includes a relational database management system (RDBMS), where information is maintained in one or more database tables for easy, efficient storage and retrieval. In addition to database tables, the system provides "design documents" which allow a user to customize how his or her data are presented, including formats which are not tabular. Design documents can also link together different tables, so that information stored in separate tables appears to the user to come from one place.

Drawing Description Text (3):
FIG. 1B is a block diagram of a software system of the present invention, which includes operating system, application software, relational database management system, and user interface components.

Detailed Description Text (7):
Database Management System (DBMS): System that controls the organization, storage, and retrieval of information in a database.

Detailed Description Text (13):
header: Typically the first data in a file, a header stores identity, status, and other data of a file.

Detailed Description Text (20):
row: Physically, a row is usually a record in a data file. Logically, a row is one horizontal member of a table: a collection of fields.

Detailed Description Text (25):
The following description will focus on the presently preferred embodiment of the present invention, which is operative in the Microsoft.RTM. Windows environment. The present invention, however, is not limited to any particular one application or any particular windows environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously applied to a variety of system and application software, including database management systems, wordprocessors, spreadsheets, and the like. Moreover, the present

invention may be embodied on a variety of different platforms, including Macintosh, UNIX, NeXTSTEP, and the like. Therefore, the description of the exemplary embodiments which follows is for purposes of illustration and not limitation.

Detailed Description Text (30):
Illustrated in FIG. 1B, a computer software system 150 is provided for directing the operation of the computer system 100. Software system 150, which is stored in system memory 102 and on disk memory 107, includes a kernel or operating system (OS) 140 and a windows shell 145. One or more application programs, such as application software 125 or one or more windows application software 151, 153, 155, may be "loaded" (i.e., transferred from storage 107 into memory 102) for execution by the system 100. As shown, windows application software includes a Relational Database Management System (RDBMS) 155 of the present invention.

Detailed Description Text (31):
System 150 includes a user interface (UI) 160, preferably a Graphical User Interface (GUI), for receiving user commands and data. These inputs, in turn, may be acted upon by the system 100 in accordance with instructions from operating module 140, windows 145, and/or application modules 125, 151, 153, 155. The UI 160 also serves to display the results of operation from the OS 140, windows 145, and applications 125, 151, 153, 155, whereupon the user may supply additional inputs or terminate the session. Although shown conceptually as a separate module, the UI is typically provided by interaction of the application modules with the windows shell, both operating under OS 140. In a preferred embodiment, OS 140 is MS-DOS and windows 145 is Microsoft.RTM. Windows; both are available from Microsoft Corporation of Redmond, Wash. RDBMS 155 includes Paradox.RTM. for Windows Database Management System, available from Borland International of Scotts Valley, Calif.

Detailed Description Text (32):
B. Relational Database Management System

Detailed Description Text (35):
In a relational database management system, information is represented in tables. As conceptually shown in FIG. 1C, a table 170 is organized (logically) into horizontal rows (tuples) 173 and vertical columns 175, thus making it easy for a user to examine or change data. Each row or "record" contains all available information about a particular item, such as storing information about an individual person, place, or thing (depending on what the table tracks). A record for an employee, for instance, may include information about the employee's ID Number, Last Name and First Initial, Position, Date Hired, Social Security Number, and Salary. Thus, a typical record includes several categories of information, that is, each record in the table is made up of several categories of information about one specific thing.

Detailed Description Text (38):
Internally, tables may be stored by the system as a sequence of fixed-length or variable-length binary records in a single disk file. The system uses a record number as an internal counter to keep track of each record. Between the actual physical database itself (i.e., the data actually stored on a storage device) and the users of the system, therefore, a database management system or DBMS provides a software cushion or layer. Because the DBMS shields the database user from knowing or even caring about underlying hardware-level details, the system manages record numbers automatically, with precautions taken so a user cannot change them directly. Thus, all requests from users for access to the data, including requests to retrieve, add, or remove information from files, are processed by the RDBMS without the user's knowledge of underlying system implementation.

Detailed Description Text (77):
The creation of information tables and design documents will be illustrated for a small sales order-management database, which is sophisticated enough to demonstrate

the elements of creating a relational model, but is sufficiently simple for clarity. The database includes a plurality of information tables 375 as shown in FIG. 3F. It includes a Customer table (CUSTOMER.DB) for storing customer data, and includes an Orders table (ORDERS.DB) for storing information about each order made by a customer. As shown, each of these two tables includes a common field: Customer No. The two tables may, therefore, be linked through this common field. Moreover, to maintain integrity of the Orders table, no order should be accepted for a customer which does not exist; in other words, the Orders table is preferably dependent on the Customer table (in a child-to-parent or detail-to-master relation).

Detailed Description Text (98):
Free-form ("Memo") Field Storage and Free-Space Management

Detailed Description Text (99):
Referring now to FIGS. 7A-D, a method of the present invention for free-form storage and management will be described. As shown in FIG. 7A, a database table 700 includes tuples or records, such as records 711, 713, 715. In addition to storing the aforementioned field types (e.g., alphanumeric, number, date, and the like), a database may store free-form or "memo" information for each record. As shown in the figure, each record includes a pointer or link which connects the record to a particular memo block stored in a separate memo file, such as the memo file 720.

Detailed Description Text (100):
Conventional DBMS systems store memo information by allocating fixed-length storage blocks. In dBASE III.RTM., for instance, a table of database records would store memo information in an accompanying memo file comprising 512-byte storage blocks. Unfortunately for the user, however, a record having only 40 bytes of memo information would require as much storage space as one having 500 bytes. Moreover, conventional systems include no free-space management which would allow reclamation of storage space which has been freed (e.g., after its corresponding database table record has been deleted).

Detailed Description Text (102):
Following the header block 721 is a free list block 723. Free list 723 stores free list entries which allow the system to perform space management. Each free list (e.g., list 723) is chained (points) to another free list (e.g., free list 725); the pointer of the last free list is set to NULL. In this fashion, there is no preset limit to the size which the memo file 720 may grow.

Detailed Description Text (105):
The actual storage blocks themselves are available in two different sizes: large and small. Large data block 730, for instance, is a data block having a size which is a multiple of the native block size (i.e., multiple of 4K), that is sufficient for a particular storage allocation. For instance, if a data object requires 4 megabytes of storage, then a large block of 1000-4K blocks would be allocated accordingly. In a preferred embodiment, these would be allocated consecutively on the storage media, so that the data object residing in the large block may be read after a single disk seek operation. Moreover, this allows application clients to treat the data as an object residing at a particular disk region, without concern for underlying space management considerations.

Detailed Description Text (106):
A small block, such as block 740, is available for sub-allocation. Shown in further detail in FIG. 7C, small block 740 includes two portions: an index 741 and a data storage region 743. To provide sub-allocation, the index includes pairs of pointer/size entries which point to a region in the storage which has an available size associated with it. In a preferred embodiment, the small block sub-allocation is provided in units of 16-byte paragraphs. To allocate storage for 30 bytes, for instance, two 16-byte paragraphs (i.e., 32 bytes total) would be allocated from

unused space 745. The corresponding index size entry would be set equal to 2 (i.e., two storage paragraphs). The corresponding pointer entry would be set equal to an offset within the data region 743, expressed in paragraph units.

Detailed Description Text (107):
Within a small block, there may exist a significant aggregate of unused space. Consider the following example. If a 4K allocation is required, for instance, the system searches in the free list among large blocks for a 4K allocation space; this may require a larger block to be "carved up" into a 4K block, with a remainder. In a preferred embodiment, this space is made available through the free lists. If a 30-byte allocation is required, the system searches for two 16-byte paragraphs, preferably found in small blocks. Accordingly, the small blocks are also referenced in the free lists. For efficiency, the small block includes a header 747 which stores a total consumed amount.

Other Reference Publication (5):
Lock Management Architecture, IBM Technical Disclosure Bulletin (1989) 31:125-128.

CLAIMS:

1. In a system for storing information in data records, a method for storing free-form data objects apart from the data records while maintaining an association between each data object and a corresponding data record, the method comprising:

(a) partitioning a persistent storage media into a plurality of variable-length storage regions, wherein said variable-length storage regions comprise at least one large and at least one small storage block, all said at least one small block having a uniform storage size, all said at least one large storage block having a storage size which is a multiple of said uniform storage size;

(b) storing a list of ones of said storage regions which are available for data storage;

(c) receiving a request for storing a data object associated with a particular one of the data records;

(d) selecting from said list at least one of said large storage regions for attaining space equal to or less than required for storing the data object;

(e) if additional storage is still required for storing the data object, selecting from said list at least one of said small storage regions for storing the data object for attaining space sufficient for storing the data object;

(f) storing the data object of interest in said selected one or more regions; and

(g) storing a handle to the data object in the associated data record, so that the data object can be associated with its particular data record.

2. The method of claim 1, further comprising:

(g) storing within the associated data record a data portion selected from the data object, said data portion being stored as a data field of the associated data record.

5. The method of claim 2, further comprising:

receiving a request from a user for viewing the data record associated with said data object; and

presenting to the user information from data fields of the data record together

with only said data portion selected from the data object which is stored as a data field of the associated data record.

8. The method of claim 2, further comprising:

receiving a user request for retrieving the data object; and

in response to the user request for retrieving the data object, retrieving said data object by:

reading from the associated data record said handle to the data object,

with said handle, retrieving said data object from said storage media, and

upon retrieving said data object, presenting to the user said data object in its entirety.

20. In a system for storing information in data records, a method for storing a free-form data object for a particular record, the method comprising:

(a) storing said particular record in a database table;

(b) storing said free-form data object at a storage location apart from said database table;

(c) storing within said database table as a particular data field of said particular record a copy of a portion of said free-form data object, so that a user can view a portion of said free-form data object directly from the database table without having the system actually retrieve said free-form data object from the storage location which is apart from said database table; and

(d) storing within said particular record a handle referencing where said free-form data object resides, so that the system can retrieve said free-form data object in its entirety upon demand.

25. In a system for storing information in data records on a persistent storage device, a method for storing a data object for a particular record, the method comprising:

(a) partitioning the persistent storage device for providing a plurality of variable-length storage blocks, wherein said storage blocks comprise at least one large and at least one small storage block, all said at least one small block having a uniform storage size, all said at least one large storage block having a storage size which is a multiple of said uniform storage size;

(b) maintaining a list of storage blocks of the persistent storage device which are available for storing objects, said storage blocks being stored apart from data records and having different storage sizes;

(c) selecting from said list a particular storage block comprising a large storage block having a size appropriate for attaining space equal to or more than required for storing the data object;

(d) storing the data object of interest in said particular storage block and removing said particular storage block from said list;

(e) if the data object requires less storage size than is available from said particular storage block, determining a remainder for the particular storage block and adding an entry to the list for indicating that said remainder is available for storing objects; and

(f) storing an entry in said particular record for indicating the particular storage block where said data object is stored, so that said data object is available on-demand from said particular record.

Previous Doc Next Doc Go to Doc#

L11: Entry 52 of 70 File: USPT Dec 12, 2000

US-PAT-NO: 6161105
DOCUMENT-IDENTIFIER: US 6161105 A

TITLE: Method and apparatus for multidimensional database using binary hyperspatial code

DATE-ISSUED: December 12, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Keighan; Edric	Quebec			CA
Vretanos; Panagiotis A.	Toronto			CA
Galluchon; Michael	Quebec			CA
Varma; Herman P.	Nova Scotia			CA

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Oracle Corporation	Redwood Shores	CA			02

APPL-NO: 08/ 827987 [PALM]
DATE FILED: October 2, 1996

PARENT-CASE:
This application is a continuation of Ser. No. 08/342,922, filed Nov. 21, 1994, now abandoned.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/100; 707/102, 707/104
US-CL-CURRENT: 707/100; 707/102, 707/104.1

FIELD-OF-SEARCH: 707/102, 707/104, 707/100

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4555771	November 1985	Hayashi	707/1
☐	4788538	November 1988	Klein et al.	345/418
☐	4794461	December 1988	Roberts et al.	358/261.3

☐	5257365	October 1993	Powers et al.	707/100
☐	5261032	November 1993	Rocchetti et al.	345/441
☐	5359724	October 1994	Erle	707/205
☐	5414780	May 1995	Carnahan	382/276
☐	5446806	August 1995	Ran et al.	382/240
☐	5647058	July 1997	Agrawal et al.	707/1
☐	5701467	December 1997	Freeston	707/100

OTHER PUBLICATIONS

Varma et al., "A Data Structue for Spatio-Temporal Databases", International Hydrographic Review, Monaco, LXVII(1), Jan. 1990.
Hsieh et al., "A Conversion and Management System for Parcel Maps", IEEE Comput. Soc. Pres., Nov. 1994.
Beng Chin Ooi, Ken J. McDonnell, Ron Sacks-Davis, "Partial kd-Tree: An Indexing Mechanism for Spatial Database", Dept. of Computer Science, Monash University, Victoria Australia; Dept. of Computing, Royal Melbourne Institute of Technology, 1987.
Yutaka Oshwawa, Masao Sakauchi, "A New Tree Type Data Structure with Homogeneous Nodes Suitable for a Very Large Spatial Database", Institute of Industrial Science, University of Tokyo, 1990.
Max J. Egenhoffer and Robert D. Franzosa, "Point-Set Topological Spatial Relations", Int. J. Geographical Information Systems, 1991, vol. 5, No. 2, 161-174.

"Implementation of HHCodes for use with hydrographic data", H. Iversen, Norwegian Hydrographic Service (NHS), http://www.statkart.no/nlhdb/hhimpl.htm, last updated Feb. 5, 1998.
"What are HH-codes and how can they be used to store hydrographic data?", H. Iversen, Norwegian Hydrographic Service (NHS), http://www.statkart.no/nlhdb/iveher/, last updated Jan. 19, 1998.
Applications of Spatial Data Structures; Computer Graphics, Image Processing, and GIS, H. Samet, University of Maryland, Addison-Wesley Publishing Company, 1990, pp. 1-15 and 174-175.
Hanan Samet, Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS, 1990, pp. 30-41 and 174-181.

ART-UNIT: 277

PRIMARY-EXAMINER: Breene; John

ASSISTANT-EXAMINER: Robinson; Greta L.

ATTY-AGENT-FIRM: McDermott, Will & Emery

ABSTRACT:

An improved database data structure and datatype is disclosed for storing, manipulating and accessing multidimensional spatial data in a database. Binary helical hyperspatial code (HH CODE) is used to represent data of N dimensions. The binary HH CODE data structure maintains the dimensional organization of multidimensional data within the data itself. Spatial data is stored using BH code which is modeled as a N-tree structure derived using recursive decomposition. A high water mark is set as the upper limit for data volume which may be stored in any one partition. As data stored in a partition exceeds the high water mark, the data is decomposed into child partitions such that no partition data stores exceed

the high water mark. If the high water mark is exceeded, additional child partitions are automatically created and the parent table is not retained. A data structure is defined which represents the partitioned tables and BH code values. Appropriate attributes are associated with each of the BH code values which may represent non-spatial data such as temperature, salinity, or cosmic ray flux. Methods and apparatus are also provided to apply teachings of binary HH CODE to line segments and topology.

69 Claims, 30 Drawing figures

Previous Doc Next Doc Go to Doc#

L11: Entry 52 of 70 File: USPT Dec 12, 2000

DOCUMENT-IDENTIFIER: US 6161105 A
TITLE: Method and apparatus for multidimensional database using binary hyperspatial code

Brief Summary Text (3):
The present invention relates to methods and apparatus for storing and manipulating multidimensional data, and more particularly, the present invention relates to database structures and methods wherein spatial data is managed within the framework of a relational database system.

Brief Summary Text (6):
The storage and manipulation of spatial data in a traditional relational database management system presents a variety of problems. Since the relational database management system ("RDBMS") is unidimensional, each value relating to a multidimensional point must be maintained in a separate column within the database. By maintaining each value in a separate column, the organization of the spatial data is not maintained. In addition, searching for data points in a given multidimensional space requires a significant amount of computation using a relational database structure. In the case of very large relational databases used to store spatial data, the column by column range search required to locate and manipulate spatial data is computationally intensive and relatively slow.

Brief Summary Text (7):
Attempts to resolve the problem of storing and managing spatial data in relational database systems have resulted in hybrid solutions. From the perspective of the relational database model, the use of relational database systems for spatial data is inefficient. As shown in FIG. 1, one hybrid approach includes two side-by-side database engines, namely, a relational database management system 50 and a spatial database system 55. The RDBMS 50 maintains attribute data which is non-spatial. An example of attribute data is the value of temperature or ocean salinity for a three dimensional point in the Pacific Ocean. Another example of non-spatial attribute data is the cosmic ray flux through a point in space between the Earth and Mars. As illustrated in FIG. 1, the RDBMS 50 is linked to a spatial database 55 which references or links the corresponding spatial data through spatial indexes 57. One example of an existing hybrid system is a RDBMS manufactured by Oracle Corporation linked to a spatial database referred to as ARC/Info. The Oracle-ARC/Info hybrid database is used to store and manage cartographic projections and similar spatial data in conjunction with attributes related to the spatial data.

Brief Summary Text (11):
As will be described, the present invention is an improvement to the original HH Code data structure disclosed in the Varma paper. The present invention's implementation and utilization of binary hyperspatial ("BH code") overcomes the inherent limitations of prior art RDBMS in efficiently storing, manipulating and retrieving spatial data. The present invention's database method and apparatus provides for the seamless handling of both spatial and non-spatial data in the same database, and constitutes a fundamental improvement in the field of database structures and management.

Brief Summary Text (13):

The present invention discloses an improved multidimensional database wherein spatial data is managed within the framework of a relational database system. The present invention utilizes binary hyperspatial code (BH code) which is modeled as an N-dimensional tree structure, and is derived using a recursive decomposition technique. A universe of data is recursively decomposed to achieve a desired level of resolution which corresponds to the decimal precision of data. A point is conceived as residing in a region, which in the case of three dimensions results in the region having the shape of a cube. N dimensions may be represented using the present invention's binary BH code data structure. The BH code data structure of the present invention linearizes multiple dimensions into a single BH code value. The present invention's use of BH code maintains the spatial organization of the data as well as each dimension, thereby composing the BH code into a single linearized data structure. Each of the BH code values may have up to M attributes which relate to the BH code data. These attributes are user defined and correspond to physical characteristics such as temperature, cosmic ray flux, salinity and the like, or may correspond to non-physical data such as money, interest, customer names and lists, etc. Spatial data maintained using the binary BH code data structure maintains the spatial organization of the data independent of a formal index. A multidimensional table is a logical table referred to as a partitioned table which is comprised of one or more partitions or tables. Each partition represents a unique bounded N dimensional space, wherein all data in one partition exists within the same bounded region of space.

Brief Summary Text (16):
The present invention further provides methods for the utilization of binary BH code to represent, store, manipulate and access two dimensional lines and topology data. In the case of lines, a line segment is represented as a four dimensional BH code value. The loading of a line segment into the database of the present invention is analogous to the operation for the loading of point data or region data. The lines are loaded into the database using a distinct multidimensional table. A high water mark is set, such that child partitions are automatically created using the present invention's automatic data partitioning scheme. A method for accessing line data stored in the database is further provided which includes the definition of partition shapes to which a line may correspond. A partition list is created and the partition shapes are decoded. A determination is made as to whether or not the partition shapes overlap regions defined by the user. A list of partitions which overlap are then compiled and data records that are within the defined region are determined. Those records defined by BH code values disposed within the user defined region are reported to the user. Methods are also disclosed herein for management of more complex spatial objects representing topology.

Drawing Description Text (2):
FIG. 1 illustrates a prior art hybrid system in which a relational database management system and a spatial database are interconnected through indexes.

Detailed Description Text (55):
Mathematically, a line is comprised of two end points, including a first point (X.sub.1, Y.sub.1) and a second point (X.sub.2, Y.sub.2) joined by a line segment (see FIG. 20(a)). In accordance with the teachings of the present invention, a line segment is represented as a four dimensional BH CODE. The value of each of the four points X.sub.1, Y.sub.1, X.sub.2, Y.sub.2 are represented as a four dimensional BH CODE, where dimension 1 (DIM1) is equal to X.sub.1, dimension 2 (DIM2) is comprised of Y.sub.1, dimension 3 (DIM3) corresponds to X.sub.2, and dimension 4 (DIM4) is comprised of Y.sub.2 (see FIG. 20(b)). Thus, a four dimensional entity in the present invention's spatial database represents a two dimensional line segment. The loading of line segment data into the database of the present invention is analogous to the operation previously described with reference to the loading of points. The present invention's data dictionary maintains information about partitions that are used to store lines. The lines are loaded into those partitions, and as is the case with points, when the high water mark (which is the

maximum data volume allowed in a partition) is overflowed, the present invention subdivides the partition just as is described for points. The difference is that when subdividing a two dimensional point partition, the present invention subdivides into four partitions, however, since a line is represented by a four dimensional object, up to sixteen (2.sup.4) partitions may be created for lines at every subdivision. The present invention creates these partitions only for those partitions in which data will be stored. The unused partitions do not waste memory space and disk storage, and are only inferred from the data maintained in the md$ptab table within the data dictionary.

Detailed Description Text (66):
Spatial data management requires more than the management of points and lines. Spatial data management also requires the management of more complex spatial objects. For example, a data set representing a road passing through a park is comprised of complex objects composed of primitive elements. The road is comprised of line segments and the park is comprised of line segments and/or region data. The objective of the present invention is to provide a data structure to efficiently maintain a complex object, and therefore maintain its topology. Topology of an object must be maintained to determine the relationship among the various objects.

Other Reference Publication (2):
Hsieh et al., "A Conversion and Management System for Parcel Maps", IEEE Comput. Soc. Pres., Nov. 1994.

Previous Doc Next Doc Go to Doc#

L11: Entry 53 of 70 File: USPT Nov 7, 2000

US-PAT-NO: 6144970
DOCUMENT-IDENTIFIER: US 6144970 A

TITLE: Technique for inplace reorganization of a LOB table space

DATE-ISSUED: November 7, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Bonner; Charles Roy	San Jose	CA		
Lyle; Robert William	Morgan Hill	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 09/ 322316 [PALM]
DATE FILED: May 28, 1999

PARENT-CASE:
PROVISIONAL APPLICATION This application claims the benefit of U.S. Provisional Application No. 60/101,729, entitled "IMPROVED DATABASE SYSTEM," filed on Sep. 24, 1998, by Charles R. Bonner et al., attorney's reference number ST9-98-046, which is incorporated by reference herein.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/206; 707/2, 711/170
US-CL-CURRENT: 707/206; 707/2, 711/170

FIELD-OF-SEARCH: 707/206, 707/2, 395/182.02, 711/170

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4509119	April 1985	Gumaer et al.	711/136
☐	4695949	September 1987	Thatte et al.	707/206
☐	4807120	February 1989	Courts	707/206

☐	4949388	August 1990	Bhaskaran	382/159
☐	4961134	October 1990	Crus et al.	707/8
☐	5043866	August 1991	Myre, Jr. et al.	707/202
☐	5088036	February 1992	Ellis et al.	707/206
☐	5136706	August 1992	Courts	707/206
☐	5222235	June 1993	Hintz et al.	707/101
☐	5247672	September 1993	Mohan	711/152
☐	5261088	November 1993	Baird et al.	707/206
☐	5291583	March 1994	Bapat	395/705
☐	5295188	March 1994	Wilson et al.	380/30
☐	5396623	March 1995	McCall et al.	707/101
☐	5408654	April 1995	Barry	707/101
☐	5416915	May 1995	Mattson et al.	711/114
☐	5418921	May 1995	Cortney et al.	711/114
☐	5418940	May 1995	Mohan	714/5
☐	5435004	July 1995	Cox et al.	707/205
☐	5452299	September 1995	Thessin et al.	370/260
☐	5455944	October 1995	Haderle et al.	707/202
☐	5517641	May 1996	Barry et al.	707/101
☐	5560003	September 1996	Nilsen et al.	707/206
☐	5566329	October 1996	Gainer et al.	707/4
☐	5579499	November 1996	Fecteau et al.	711/209
☐	5579515	November 1996	Hintz et al.	707/7
☐	5630093	May 1997	Holzhammer et al.	711/115
☐	5666560	September 1997	Moertl et al.	710/68
☐	5684986	November 1997	Moertl et al.	707/101
☐	5687343	November 1997	Fecteau et al.	711/202
☐	5721915	February 1998	Sockut et al.	707/200
☐	5727197	March 1998	Burgess et al.	707/2
☐	5732402	March 1998	Lehman	707/205
☐	5737601	April 1998	Jain et al.	707/201
☐	5742806	April 1998	Reiner et al.	707/3
☐	5742810	April 1998	Ng et al.	707/4
☐	5748952	May 1998	Chadha et al.	707/2
☐	5758357	May 1998	Barry et al.	707/202
☐	5761667	June 1998	Koeppen	707/101
☐	5857210	January 1999	Tremblay et al.	707/206
	5909540	June 1999	Carter et al.	395/182.02

☐				
☐	5963982	October 1999	Goldman	711/170

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
8-167852	June 1996	JP	

OTHER PUBLICATIONS

IBM Technical Disclosure Bulletin, "Fine Granularity Locking to Support High Data Availability in a Client/Server Database Management System," vol. 38, No. 02, pp. 143-145, Feb. 1995.
Joon Seek Kim, et al., "Mapping Parameter Estimation Using Integral Projections And Segmented Moving Objects in Object-Oriented Analysis-Synthesis Coding," Optical Engineering, vol. 35, No. 1, pp. 156-165, Jan. 1996.
MJ Carey, et al., "Object And File Management in the EXODUS Extensible Database System," Proceedings of Very Large Data Bases. Twelfth International Conference on Very Large Data Bases, Kyoto, Japan, pp. 91-100, Aug. 25-28, 1986.
ML McAuliffe, et al., "Towards Effective and Efficient Free Space Management," 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, Jun. 4-6, 1996.
C. Mohan, "Disk Read-Write Optimizations and Data Integrity in Transaction Systems Using Write-Ahead Logging," Proceedings of the Eleventh International Conference on Data Engineering (Cat. No. 95CH35724), Taipei, Taiwan, Mar. 6-10, 1995.
Ki Sik Pang, et al., "An Efficient Recovery Scheme For Large Data in Multimedia DBMS," Journal of the Korea Information Science Society, vol. 22, No. 2, pp. 206-217, Feb. 1995.
C. Mohan, et al., "Algorithms For Flexible Space Management in Transaction Systems Supporting Fine-Granularity Locking," Advances in Database Technology--EDBT '94. 4th International Conference on Extending Database Technology, Cambridge, UK, Mar. 28-31, 1994.
Martin Marshall, "Time Warner Big on Oracle Objects. (Testing Oracle 8's Ability to Move Large Object Blocks)," (Company Operations), (Brief Article), CommunicationsWeek Issue: n676, pp. 1-3, Aug. 11, 1997.
HweeHwa Pang, "Tertiary Storage in Multimedia Systems: Staging or Direct Access?", Multimedia Systems, vol. 5, Issue: 6, pp. 386-399, Dec. 1, 1997.
Dr. Michael Stonebraker, "The Empire Strikes Back: DB2 Universal Database," http://www.oreview.com/9704side.htm, pp. 1-7, 1997.
GH Sokut, "A Method For On-Line Reorganization of a Database," IBM Systems Journal, vol.. 36, No. 3 pp. 411-436, 1997.
H. Koide, et al., "A New Memory Allocation Method For Shared Memory Multiprocessors With Large Virtual Address Space," Concurrency:Practice and Experience, vol. 9, No. 9, pp. 897-914, Sep. 1997.
IBM Technical Disclosure Bulletin, "Method For Storing Large Objects in a Relational Database," vol. 35, No. 4A, pp. 72-75, Sep. 1992.
IBM Technical Disclosure Bulletin, "Reorganization Flags For Table Indexes," vol. 35, No. 5, pp. 156-157, Oct. 1992.
IBM Technical Disclosure Bulletin, "Technique to Allow DB2 Utilities and Commands to Run While SQL Applications Have a Table Space Locked," vol. 36, No. 09A, pp. 499-501, Sep. 1993.
IBM Technical Disclosure Bulletin, "Spanning Temporary Reorg Files," vol. 36, N. 06A, p. 159, Jun. 1993.
IBM Technical Disclosure Bulletin, "Segmented Relational Database Tables," vol., 38, No. 07, pp. 219-220, Jul. 1995.
IBM Technical Disclosure Bulletin, "Mapping a Relational Database to a Hierarchical

File System," vol. 38, No. 10, pp. 309-311, Oct. 1995.

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Rones; Charles L.

ATTY-AGENT-FIRM: Pretty, Schroeder & Poplawski, P.C.

ABSTRACT:

A apparatus, apparatus, and article of manufacture for a computer implemented inplace reorganization system. Data in a database is stored on a data storage device connected to a computer is reorganized. Large object data that needs to be reorganized within a table space is identified. One or more chunks in the table space are allocated to the identified large object data. In particular, a combination of full and partial chunks of space within the table space are allocated to contain the reorganized large object data. Then, the large object data is moved into the allocated chunks to reorganize the large object data inplace. Furthermore, the free space at the end of the table space may be reclaimed.

72 Claims, 14 Drawing figures

Previous Doc Next Doc Go to Doc#

L11: Entry 53 of 70 File: USPT Nov 7, 2000

DOCUMENT-IDENTIFIER: US 6144970 A
TITLE: Technique for inplace reorganization of a LOB table space

Brief Summary Text (16):
Databases are computerized information storage and retrieval systems. A Relational Database Management System (RDBMS) is a database management system (DBMS) which uses relational techniques for storing and retrieving data. Relational databases are organized into tables which consist of rows and columns of data. The rows are formally called tuples or records. A database will typically have many tables and each table will typically have multiple tuples and multiple columns. Tables are assigned to table spaces. A table space is associated with direct access storage devices (DASD), and, thus, tables are stored on DASD, such as magnetic or optical disk drives for semi-permanent storage.

Brief Summary Text (17):
A table space can be a system managed space (e.g., an operating system file system) or a database managed space. Each table space is physically divided into equal units called data pages or pages. Each page, which typically contains 4K bytes, holds one or more rows of a table and is the unit of input/output (I/O). The rows of a table are physically stored as records on a page. A record is always fully contained within a page and is limited by page size. As users move towards working with image data and other large data objects, storing data in conventional records becomes difficult.

Brief Summary Text (19):
Traditionally, a RDBMS stored simple data, such as numeric and text data. In a traditional RDBMS, the underlying storage management has been optimized for simple data. More specifically, the size of a record is limited by the size of a page, which is a fixed number (e.g., 4K) defined by a computer developer. This restriction in turn poses a limitation on the length of columns of a table. To alleviate such a restriction, most computer developers today support a new built-in data type for storing large objects (LOBs). Large objects, such as image data, typically take up a great deal of storage space.

Brief Summary Text (20):
If there is a clustering index defined, the DBMS will attempt to insert the record in the same order as the clustering keys. Maintaining data records in the clustering key order enables more efficient data retrieval when the clustering index is used to retrieve a set of records within a key range.

Brief Summary Text (22):
Unloading and reloading a table space has several problems. First, at least twice as much DASD space as is required for the table space is required to perform a non-inplace reorganization. Generally, LOB table spaces are used to store LOB values. Thus, LOB table spaces are very large and management of the work space is a usability problem.

Detailed Description Text (5):
Operators of the computer system 102 use a standard operator interface 108, such as IMS/DB/DC.RTM., CICS.RTM., TSO.RTM., OS/390.RTM., ODBC.RTM. or other similar

interface, to transmit electrical signals to and from the computer system 102 that represent commands for performing various search and retrieval functions, termed queries, against the databases. In the present invention, these queries conform to the Structured Query Language (SQL) standard, and invoke functions performed by Relational DataBase Management System (RDBMS) software.

Detailed Description Text (8):
As illustrated in FIG. 1, the DB2.RTM. system for the OS/390.RTM. operating system includes three major components: the Internal Resource Lock Manager (IRLM) 110, the Systems Services module 112, and the Database Services module 114. The IRLM 110 handles locking services for the DB2.RTM. system, which treats data as a shared resource, thereby allowing any number of users to access the same data simultaneously. Thus concurrency control is required to isolate users and to maintain data integrity. The Systems Services module 112 controls the overall DB2.RTM. execution environment, including managing log data sets 106, gathering statistics, handling startup and shutdown, and providing management support.

Detailed Description Text (9):
At the center of the DB2.RTM. system is the Database Services module 114. The Database Services module 114 contains several submodules, including the Relational Database System (RDS) 116, the Data Manager 118, the Buffer Manager 120, the Inplace Reorganization System 124, and other components 122 such as an SQL compiler/interpreter. These submodules support the functions of the SQL language, i.e. definition, access control, interpretation, compilation, database retrieval, and update of user and system data. The Inplace Reorganization System 124 works in conjunction with the other submodules to reorganize data inplace.

Detailed Description Text (10):
The present invention is generally implemented using SQL statements executed under the control of the Database Services module 114. The Database Services module 114 retrieves or receives the SQL statements, wherein the SQL statements are generally stored in a text file on the data storage devices 104 and 106 or are interactively entered into the computer system 102 by an operator sitting at a monitor 126 via operator interface 108. The Database Services module 114 then derives or synthesizes instructions from the SQL statements for execution by the computer system 102.

Detailed Description Text (19):
One of the advantages of rechunking is that LOB pages are moved to gain effective prefetch (i.e., retrieval of LOB pages just prior to their use). In particular, a LOB map identifies the pages to which a LOB is allocated. The pages need not be contiguous. Using the LOB map, the data manager system 118 identifies data pages for prefetch. For example, if a LOB is allocated to page 20, 1000, and 50, the PFPL would contain entries for 20, 1000, and 50. When LOB pages are to be prefetched, the data manager system 118 uses the PFPL to prefetch LOB pages. Similarly, the inplace reorganization system 124 can use the PFPL to prefetch LOB low-level space map pages. The PFPL is explained in further detail in the above cross-referenced application entitled "AN OPTIMIZED TECHNIQUE FOR PREFETCHING LOB TABLE SPACE PAGES". For each LOB, effective prefetching is attained by placing non-contiguous LOB pages adjacent to one another in "chunks". The chunks allocated to the LOB do not have to be contiguous for prefetch to work effectively.

Detailed Description Text (25):
The LOB table space 206 contains an auxiliary table 210. The inplace reorganization system 124 requires that users define an auxiliary table 210 within the LOB table space 206 to contain the actual LOB values. The auxiliary index 208 is created on the auxiliary table 210 in index space 216. The data manager 118 has been extended to find LOB values. In particular, the data manager 118 uses the auxiliary index 208 to quickly find the LOB values for a specific row. In particular, the auxiliary index contains keys 214, which indicate the first LOB map page, such as LOB Map

Pagel 212. The first LOB map page acts as a directory to the LOB map and LOB pages of a LOB and assists with accessing the LOB data. In addition to LOB Map pages, such as LOB Map Pagel 212, the auxiliary table 210 contains LOB low-level space map pages, such as LOB Low-Level Space Map Page 1 218. LOB low-level space map pages assist in allocating and deallocating LOB pages. A high-level space map identifies the low-level space map pages.

Detailed Description Text (31):
The free space list is used to manage the free space efficiently when allocating and deallocating pages. For each unique LOB space map, and for each chunk, free space list is used to identify the number of free pages within the chunk. The LOB low-level space map page list is used to locate LOB low-level space map pages and chunks when deallocating space. The free space page list is used to locate a chunk with the required amount of free pages. The singly linked lists may share common elements to facilitate maintenance of the lists.

Other Reference Publication (1):
IBM Technical Disclosure Bulletin, "Fine Granularity Locking to Support High Data Availability in a Client/Server Database Management System," vol. 38, No. 02, pp. 143-145, Feb. 1995.

Other Reference Publication (3):
MJ Carey, et al., "Object And File Management in the EXODUS Extensible Database System," Proceedings of Very Large Data Bases. Twelfth International Conference on Very Large Data Bases, Kyoto, Japan, pp. 91-100, Aug. 25-28, 1986.

Other Reference Publication (4):
ML McAuliffe, et al., "Towards Effective and Efficient Free Space Management," 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, Jun. 4-6, 1996.

Other Reference Publication (7):
C. Mohan, et al., "Algorithms For Flexible Space Management in Transaction Systems Supporting Fine-Granularity Locking," Advances in Database Technology--EDBT '94. 4th International Conference on Extending Database Technology, Cambridge, UK, Mar. 28-31, 1994.

CLAIMS:

22. The method of claim 20, wherein the step of identifying further comprises the step of tracking free space encountered while searching low-level space map pages to identify large objects that need to be reorganized and wherein the step of determining whether the last large object can be moved further comprises the step of calculating whether the last large object fits in the tracked free space while leaving a portion of the free space unused.

46. The apparatus of claim 44, wherein the means for identifying further comprises the means for tracking free space encountered while searching low-level space map pages to identify large objects that need to be reorganized and wherein the means for determining whether the last large object can be moved further comprises the means for calculating whether the last large object fits in the tracked free space while leaving a portion of the free space unused.

70. The article of manufacture of claim 68, wherein the step of identifying further comprises the step of tracking free space encountered while searching low-level space map pages to identify large objects that need to be reorganized and wherein the step of determining whether the last large object can be moved further comprises the step of calculating whether the last large object fits in the tracked free space while leaving a portion of the free space unused.

Previous Doc Next Doc Go to Doc#